프로덕트 라인 기반의 모바일 소프트웨어 개발 프로세스

김 행 곤[†] · 손 이 경[††]

요 약

유비쿼터스 컴퓨팅은 매우 광범위한 기술 분야에 적용될 수 있고 완벽한 사용자 요구를 필요로 하며, 많은 시나리오와 기술들을 포함하고 있으므로 이러한 요구를 충족시켜주는 새로운 소프트웨어 개발 툴과 방법론이 필수적이다. 이를 위한 새로운 기술로써 소프트웨어 프로덕트 라인은 공통의 유사한 기능을 가지고 있는 소프트웨어 제품 혹은 소프트웨어 시스템 집합으로 특정 영역의 시장과 용도의 요구사항에 따라 재사용 가능한 아키텍처 및 컴포넌트를 구성함으로써 생산성과 품질을 향상시킬 수 있다. 특히, 시스템을 분할하고 구조화하여 시스템의 성능과 효율성을 향상시킬 수 있는 소프트웨어 아키텍처 개념이 중요시 되면서 아키텍처의 개발과 평가에 대한 체계적인 연구가 필요하다. 본 논문에서는 CBD(Component Based Development)를 기반한 소프트웨어 프로덕트 라인(PLD : Product Line based Development)을 도입하여 모바일 비즈니스 도메인에 적합한 모바일 응용 시스템 아키텍처(MASA: Mobile Application System Architecture)를 제시한다.

키워드 : 프로덕트 라인, 소프트웨어 아키텍처, 모바일 응용 시스템

Product Line Development Process for Mobile Software based on Product Line

Haeng-Kon Kim[†] · Lee-Kyeong Son[††]

ABSTRACT

Ubiquitous computing spans a very broad range of technologies and needs very complicated user's requirements. There are many scenarios and technologies involved in ubiquitous computing. We need new software development tools and methodology to meet the requirements. A software product line is one of promising new technology for it. A software product line is a set of software intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets. Software architecture-based development is the exploration and maturation of the role of software architecture in the product line life cycle. In this thesis, we identify the foundational concepts underlying software product lines and the essential activities to develop the mobile application systems. So, we define, design, and implement the Mobile Application System Architecture(MASA) that includes the development process for applying into mobile business domain and encompass scoping and gathering requirements for the product line based on Component Based Development(CBD).

Key Words : Product Line, Software Architecture, Mobile Application System

1. 서 론

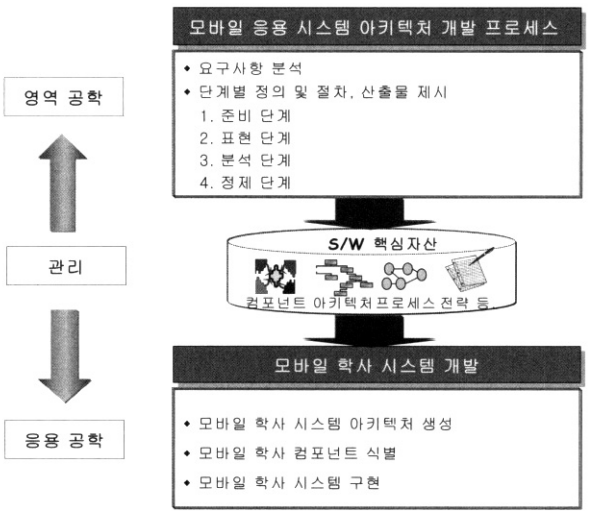
최근 유비쿼터스 사회의 도래로 모바일 비즈니스에 대한 서비스 수요의 증대와 다양한 사용자 요구로 인해 소프트웨어 산업은 빠르게 변화하고 있다. 또한 다양한 비즈니스 영역에서의 새로운 요구사항에 대한 변경과 적용은 시스템의 성공뿐만 아니라 프로젝트의 효율성 및 생산성에도 영향을 주기 때문에 요구사항과 소프트웨어 특성을 적시에 반영할 수 있는 새로운 소프트웨어 개발 기술이 필수적이다[1, 2, 3]

기존 소프트웨어 생산기술들은 소프트웨어 개발 생산성 향상에 제한적이므로 컴포넌트 기반 개발(CBD: Component Based Development) 기술을 확장하여 소프트웨어 아키텍처, 소프트웨어 컴포넌트 등 핵심자산들을 체계적으로 자원화하고 조직적으로 재사용하는 소프트웨어 생산기술인 프로덕트 라인 개발(PLD : Product Line Development) 기술이 필요하다.

프로덕트 라인 개발은 소프트웨어 핵심자산의 체계적인 개발과 관리 및 조직적인 재사용을 통해 다양한 요구사항을 쉽고, 빠르게 반영할 수 있는 맞춤형 소프트웨어 제품 생산 기술이다. 특히 소프트웨어 핵심자산 중에 아키텍처는 품질 속성을 중심으로 아키텍처 설계 결정을 제시하고 분석함으로써 풍부한 재사용 가능성을 제공한다. 또한 핵심자산의 보급과 유통, 그리고 사용을 위한 핵심자산 저장소의 개발과 적용은 다양한 프로덕트의 생산성과 품질 향상 및 시간과 비용을 절감할 수 있다[4, 5, 6].

본 논문에서는 모바일 응용 시스템의 개발 기간과 비용을

† 종신회원 : 대구가톨릭대학교 컴퓨터공학과 교수
†† 준 회 원 : 대구가톨릭대학교 컴퓨터정보통신공학부 누리(NURI)전담교수
논문접수 : 2005년 2월 1일, 심사완료 : 2005년 3월 23일



(그림 1) 연구 내용

줄이고 소프트웨어의 품질을 보장하며 사용자들의 다양한 요구사항을 쉽고 빠르게 반영하기 위해 CBD 기반의 프로덕트 라인을 도입하여 모바일 응용 시스템 개발에 적용하였다.

(그림 1)에서와 같이 모바일 응용 시스템 개발에 프로덕트 라인 개발을 적용하여 영역 공학과 관리, 응용 공학을 모두 포함하는 모바일 프로덕트 라인 개발 프로세스를 정의한다. 영역 공학 단계에서는 모바일 시스템의 요구사항을 분석하여 모바일 응용 시스템 도메인에 적합한 아키텍처 개발 프로세스를 제안한다. 제안된 모바일 응용 시스템 아키텍처(MASA : Mobile Application System Architecture)는 준비, 표현, 분석, 정제 단계로 구성되며 각 단계마다 세부적인 작업 절차와 산출물을 제시한다. 이들 세부 작업을 수행함에 따라 품질 속성 중심의 체계적인 아키텍처 개발이 가능하다.

관리 단계에서는 영역 공학에서 나타난 다양한 산출물들과 프로세스를 핵심자산 저장소를 통해 저장하고 관리한다. 마지막으로 응용 공학 단계에서는 개발된 모바일 응용 시스템 아키텍처를 대학 내 학사 시스템에 적용함으로써 모바일 학사 시스템 아키텍처(MESA : Mobile Educational System Architecture)를 생성하며 컴포넌트를 식별하고 구현한다. 이렇게 생성된 컴포넌트 또한 핵심자산 저장소에 저장되고 또 다른 재사용을 위해 관리된다.

2. 관련 연구

2.1 프로덕트 라인

프로덕트 라인은 공통의 유사한 기능을 가지고 있는 소프트웨어 시스템 집합으로 핵심자산 개발과 프로덕트 개발 측면을 포함하고 있다. 프로덕트 라인 아키텍처는 연관된 프로덕트 집합과 조직에 의해 개발된 시스템을 위한 공통 아키텍처로서 생산성과 품질 향상에 이바지한다[7].

독일의 IESE 연구소에 의해 제시된 KobrA는 PluSE 프레임워크를 프로덕트 라인으로 실현한 것이다. 핵심 개념은 프로덕트 패밀리의 공통적인 프레임워크 즉, 각각의 프로덕트들이 가지는 공통성과 변화성을 표현하는 컴포넌트와 이들 간의 관계성을 계층적으로 명세화한 프레임워크를 정의하고 이를 개별 프로덕트의 요구사항에 맞도록 초기화하는 것이다. 컴포넌트 및 프레임워크 기반 기술을 최대한 반영함으로써 실제적인 프로덕트 생성을 위한 구체적인 접근 방식을 제공하지만 CBD에 치중하여 새로운 프레임워크 요구의 수용을 위한 지속적인 유지보수 문제가 발생하게 되는 단점이 있다[8].

CMU SEI에서 개발된 SEI 프레임워크는 프로덕트 라인 개발과 공학적 원리를 기반으로 하고 있다. 프로덕트 라인의 개념과 프로덕트 개발 전에 고려해야 할 필수 행위 및 조직이 마스터해야 하는 실제 부분을 제시하고 소프트웨어 개발 시 프로덕트 라인 접근을 어떻게 적용하는지에 대한 기술적, 관리적 프로세스 및 지침을 제공하고 있다. 장점으로는 프로덕트 라인 관점에 대한 넓은 범위의 상세한 개요 설명을 제시하고 있으며, 효율적인 프로덕트 라인 생성에 중요하고 실제적인 기술 및 관리 부분들에 대한 설명이 이루어져 있다는 것이다. 하지만 광범위한 내용으로 프로덕트 라인으로의 구체적인 접근을 위한 지침이 미흡하다[8, 9].

2.2 모바일 시스템

현대 사회에서 인터넷을 제외하고는 개인이나 사회생활을 이야기하기 어려울 정도로 인터넷이 차지하는 비중은 상당하다고 할 수 있다. E-메일은 생활의 필수가 되었고 인터넷을 통하여 쇼핑, 예약, 은행 관련 업무뿐 아니라 각종 동영상을 실시간으로 볼 수도 있게 되었다. 여기에 언제 어디서나 통화할 수 있는 휴대폰의 등장으로 이동통신과 인터넷이 결합하여 시간과 물리적 공간의 한계를 벗어나 이동 중에 무선으로 인터넷 정보를 송·수신할 수 있게 되었다.

모바일 시스템은 모바일 기기를 이용하여 인터넷에 접속하여 이동 중에 무선으로 인터넷 정보를 송·수신할 수 있는 서비스 시스템으로서 필요할 때 편리하게 이용 가능하고 장소에 구애받지 않는다. 모바일 서비스가 주는 편리함과 자유로움은 이제 단순히 음성 통화에 국한되지 않고 데이터 전송 서비스(SMS), E-메일 서비스, 모바일 컨텐츠, 모바일 커머스, 모바일 비즈니스까지 그 범위를 넓히고 있다[10].

2.3 소프트웨어 아키텍처

소프트웨어 시스템의 규모와 복잡성이 증가함에 따라 시스템을 분할하여 구조화함으로써 시스템의 성능을 향상시키고 유지보수 비용을 절감할 수 있는 소프트웨어 아키텍처에 대한 관심이 증가하고 있다. 소프트웨어 아키텍처란 소프트웨어 컴포넌트, 이들 컴포넌트들의 가시적인 속성, 그리고 컴포넌트들 사이의 관계로 구성된 시스템의 전체적인 구조로서 시스템을 설계하고 발전시키기 위한 지침과 원리라고 정의할 수 있다. 시스템을 위한 기술적인 청사진으로서 시스템을 이해하기 위한 최선의 요소가 되며, 다양한 스테이

크홀더 간의 원활한 의사소통의 수단으로서 시스템을 바라보는 스테이크홀더들의 다양한 관점을 표현하여 서로를 이해하고 합의를 도출할 수 있게 해준다. 뿐만 아니라, 프로젝트 초기에 결정된 설계 사항을 기재해 놓은 산출물로서 시스템 품질 속성에 많은 영향을 미치며 조직을 구성하고 시스템을 설계하기 위한 전략을 세울 수 있도록 도와준다. 마지막으로 소프트웨어 아키텍처는 여러 시스템을 위해 재사용할 수 있는 시스템 모델로서 동일한 아키텍처를 사용하는 미래의 시스템을 이해하는데 도움이 된다. 아키텍처 설계자의 결정을 기록하고 아키텍처 간의 원활한 통신을 위해서는 완벽하고 명확한 문서가 필수적이다[11].

소프트웨어 품질 속성이란 시스템이 만족해야 하는 품질에 대한 요구사항을 의미하는 것으로 시스템 생명주기 초반에 완벽한 품질에 대한 요구가 결정되기 어렵지만 품질 속성을 만족하지 못하는 자원이 시스템 구축에 사용될 경우 시스템 전체가 품질에 대한 요구를 만족시키지 못하는 경우가 발생할 수 있다[12, 13]. (그림 2)는 모바일 시스템 아키텍처에서 요구되는 품질 요소로서 즉시성, 편재성, 신뢰성, 확장성, 사용성, 이식성 등이 포함되어있다.

기존 아키텍처 기반의 소프트웨어 개발 방법론 중 Galaxy 방법론은 KCSC의 업종별 아키텍처 개발·보급 사업의 일환으로 연구 개발되었으며, 프로덕트 라인 공학, RUP, Christine Hofmeister의 소프트웨어 아키텍처 뷰를 연구 배경으로 하고 있다[14].

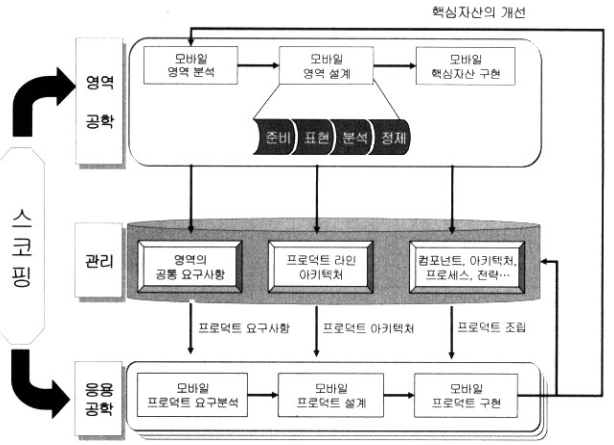
The Open Group에서 개발한 아키텍처 프레임워크인 TOGAF(The Open Group Architecture Framework)는 특정 조직의 아키텍처를 설계하고 평가, 구축하기 위한 아키텍처 프레임워크로써 기본 아키텍처와 아키텍처 개발 방법론(ADM), 자원 기반으로 구성되어 있다[15].

(그림 2) 모바일 시스템 아키텍처 품질요소

3. 모바일 프로덕트 라인 개발 프로세스

모바일 프로덕트 라인 개발 프로세스는 연관된 시스템 그룹의 아키텍처를 기반으로 공통성과 변화성 부분으로 구성되고, 특정 도메인내의 여러 애플리케이션 개발을 위해 소프트웨어 아키텍처와 컴포넌트 기반 개발의 결합으로 정의될 수 있으며, 프로세스의 정의는 기존 개발 프로세스와는 달리 영역 공학과 응용 공학의 두 가지 접근을 사용하며 이들은 각각 관리 단계에 의해 관리된다.

(그림 3)은 모바일 프로덕트 라인 개발 프로세스를 도식화한 것으로 영역 공학에서 모바일 시스템의 요구사항을 분석하고 모바일 프로덕트 라인 아키텍처 개발 프로세스를 정의하며 각 단계마다 산출물을 정의한다. 산출물들은 저장소에서 핵심자산으로 관리되며 이들 핵심자산을 재사용하여 응용 공학에서는 모바일 프로덕트인 모바일 학사 시스템을 개발한다.



(그림 3) 모바일 프로덕트 라인 개발 프로세스

3.1 모바일 응용 시스템 아키텍처 개발 프로세스

3.1.1 모바일 요구사항 분석

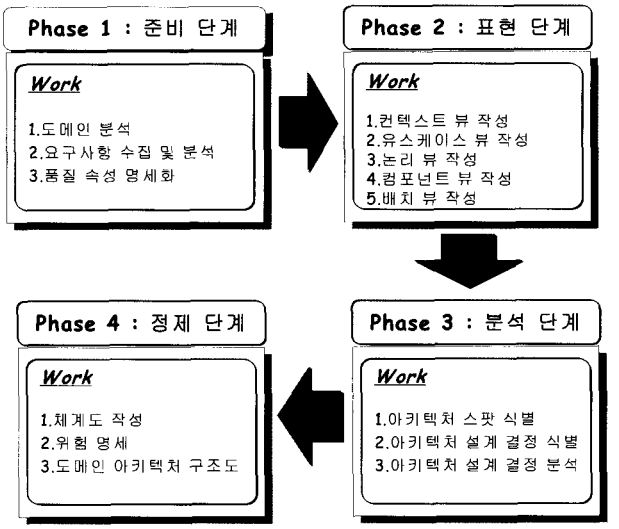
모바일 시스템은 사용자가 요구한 정보를 실시간으로 제공해주어야 하며, 언제 어디서나 사용가능하기 위해 풍부한 AP(Access Point)를 확보해야 한다. 또한 사용자별로 맞춤 서비스가 이루어지기 위해 우수한 품질과 다양한 서비스를 확보해야 한다. 이러한 모바일 시스템의 요구 사항을 만족시키기 위해 모바일 프로덕트 라인 아키텍처를 적용함으로써 양질의 핵심자산을 확보함에 따라 프로덕트의 품질을 확보하게 되고 핵심자산에 대한 재사용성이 높아질 뿐만 아니라 프로덕트의 생산성을 향상시킬 수 있게 된다.

3.1.2 아키텍처 개발 프로세스

모바일 시스템의 요구사항을 바탕으로 (그림 4)와 같이 준비단계, 표현 단계, 분석 단계, 정제 단계로 구성되는 모바일 응용 시스템 아키텍처 개발 프로세스를 제시한다.

먼저, 준비 단계에서는 목표 시스템에 대해 도메인 분석을 하고, 기능적 요구와 품질 요구를 구분하여 요구 사항 템플릿을 정의하고, 목표 시스템에서 요구하는 품질 속성에 대한 상세한 특성을 정의한다. 표현 단계에서는 준비 단계의 산출물들을 바탕으로 시스템의 전체적인 범위를 나타내는 컨텍스트 뷰를 작성하고, 4+1 뷰 모델을 적용하여 유스케이스 뷰, 논리 뷰, 컴포넌트 뷰, 배치 뷰를 표현하며 각 뷰의 설계 단계에서 각 아키텍처 뷰의 구성요소들과 그 요소들 사이의 관계를 정의한다.

분석 단계에서는 주어진 아키텍처로부터 기능적 요구에 연관된 부분을 찾아 아키텍처 스팟(architectural spots)으로 식별하고, 이로부터 품질 속성에 많은 영향을 미치는 아키



(그림 4) 모바일 응용 시스템 아키텍처 개발 프로세스

텍처 설계 결정을 식별하고 분석하여 설계 결정 템플릿을 작성한다. 정제 단계에서는 품질 속성과 아키텍처 설계 결정, 아키텍처 스팟간의 관계를 구조적으로 표현할 수 있는 체계도를 작성하고 잠재적 위험 요소를 가진 설계 결정에 대해 아키텍처 위험 명세를 기술한다. 지금까지의 결과를 바탕으로 공통 기능과 요구 기능을 중심으로 개발 환경을 고려하여 도메인 아키텍처를 정의한다.

(1) 준비 단계

가. 정의

아키텍처 기반의 소프트웨어 개발을 위한 준비 단계로서 목표 시스템에 대한 종합적인 분석이 이루어지고 조직의 요구사항을 파악하여 요구사항 템플릿을 정의하며 품질 속성에 대한 구체적인 명세를 작성하게 된다.

나. 절차

① 작업 1 : 도메인 분석

도메인 분석은 시스템의 전체적인 경계를 정의하는 단계로서 시스템의 개발 목표, 개발 필요성, 조직의 전체 업무 중 개발하고자 하는 시스템의 범위, 시스템의 주요 스테이크홀더 등을 파악한다.

② 작업 2 : 요구사항 수집 및 분석

사용자로부터 시스템이 제공하는 주요 서비스에 대한 요구 사항을 수집하여 <표 1>과 같이 요구 명세 템플릿을 작성하게 된다. 먼저 수집된 요구 사항들을 기능적 요구와 품질 요구로 분류하여 각각을 FRs 와 QRs 에 기술한다. 이때 기능적 요구는 상대적 중요도에 따라 나열하고 시스템의 공통성과 변화성을 표현하기 위해 필수적인 기능과 선택적인 기능으로 구분하여 정의한다. 품질 요구는 하나 또는 그 이상의 기능적 요구와 연관될 수 있으므로 기능적 요구와 품질 요구간의 연관 관계를 파악하여 기술하고 이들 품질 요구로부터 적절한 품질 속성을 결정하며 컨텍스트에는 각 품질 속성에 관련된 품질 요구들을 나열한다.

〈표 1〉 요구 명세 템플릿

<table>
<tr><td>품질속성</td><td colspan="3">품질 속성의 이름(QA1, QA2, . . .)</td></tr>
<tr><td rowspan="3">컨텍스트</td><td>품질 속성(QA1)</td><td colspan="2">관련 품질 요구 리스트(QR1, QR2,…)</td></tr>
<tr><td>품질 속성(QA2)</td><td colspan="2">관련 품질 요구 리스트(QR3, QR4,…)</td></tr>
<tr><td>…</td><td colspan="2">…</td></tr>
<tr><td rowspan="3">기능적 요구 (FRs)</td><td>P1 : 주요 기능의 명세1
P2 : 주요 기능의 명세2
…</td><td>1(우선순위)
2
…</td><td></td></tr>
<tr><td colspan="2">P1_M1 : 필수 기능 명세1(관련된 주요기능 표시)
P1_M2 : 필수 기능 명세2
…</td><td></td></tr>
<tr><td colspan="2">O1 : 선택 기능 명세1(관련된 주요기능 표시)
O2 : 선택 기능 명세2
…</td><td></td></tr>
<tr><td>품질 요구 (QRs)</td><td>QR1 : 품질 요구 명세1
QR2 : 품질 요구 명세2
…</td><td>관련 기능의 식별자
System
P1_M1 ...</td><td></td></tr>
</table>

③ 작업 3 : 품질 속성 명세화

준비 단계의 마지막 작업으로 목표 시스템에서 요구하는 품질 속성에 대한 특성을 <표 2>와 같이 상세히 표현할 수 있는 단계로서 식별된 품질 속성의 수만큼의 테이블이 형성된다. 먼저 요구 명세 템플릿의 품질 요구에 의해 식별된 품질 속성을 기술하고 고려될 수 있는 품질 속성의 구체적인 요인(factors)을 식별한다. 또한 이들 요인에 대해 영향을 미칠 수 있는 상황을 파악하여 원인(stimuli)에 표현하고 원인에 대한 적절한 대응 방안을 고려하여 대응(response)에 기술한다.

〈표 2〉 품질 속성 명세표

속 성	요 인	원 인	대 응
QA# : 품질 속성 이름	품질 속성에 대한 구체적 요인	품질 속성에 영향을 미칠 수 있는 상황	원인에 대한 적절한 대응
	…	…	…

다. 산출물

준비 단계의 산출물은 다음과 같다.

- 도메인 분석서
- 요구 명세 템플릿 : <표 1>
- 품질 속성 명세표 : <표 2>

(2) 표현 단계

가. 정의

도메인 분석과 요구사항 템플릿을 기반으로 컨텍스트 뷰와 UML의 4+1 뷰 모델에 해당되는 유스케이스 뷰, 논리 뷰, 컴포넌트 뷰, 배치 뷰를 작성한다. 4+1 뷰 모델은 좋은 아키텍처 문서 획득에 도움이 되는 모델로서 품질 속성이 다양하게 다루어질 수 있고 광범위한 애플리케이션에 대해

적절한 수준의 추상화를 가진 아키텍처가 기술될 수 있다.

나. 절차

① 작업 1 : 컨텍스트 뷰 작성

조직의 구성도를 중심으로 주요 작업자와 개체를 파악하고 시스템 사용자와 시스템의 기능적 요구, 작업자, 개체를 중심으로 시스템의 경계를 표현한다.

② 작업 2 : 유스케이스 뷰 작성

요구 명세 템플릿에서 정의된 시스템의 기능적 요구를 중심으로 시스템이 해야 할 행위를 명세화하는 것으로 유스케이스와 액터, 이들 간의 관계를 표현하는 작업이다.

③ 작업 3 : 논리 뷰 작성

요구 명세 템플릿과 유스케이스 뷰의 분석을 통하여 작업을 수행하는 내부 컴포넌트들의 특성 및 행위를 표현하며 시스템의 정적인 측면을 나타낸다.

④ 작업 4 : 컴포넌트 뷰 작성

개발 환경 내에서 실제 소프트웨어 모듈의 구성과 관련된 시스템의 물리적인 관점에서의 컴포넌트 구조를 표현하며 시스템을 조립하고 배포하는데 사용되는 컴포넌트와 파일들로 구성된다.

⑤ 작업 5 : 배치 뷰 작성

배치 뷰는 소프트웨어 컴포넌트들의 물리적인 배치를 표현하며 시스템을 컴퓨터와 장치의 노드로 전개시켜 서로 다른 노드들과 이 노드들 간의 연결을 나타낸다.

다. 산출물

표현단계의 다섯 가지 작업을 통해 나타나는 산출물은 다음과 같다.

- 컨텍스트 뷰
- 유스케이스 뷰
- 논리 뷰
- 컴포넌트 뷰
- 배치 뷰

(3) 분석 단계

가. 정의

주어진 기능적 요구와 아키텍처 뷰를 기반으로 품질 속성에 연관된 아키텍처 뷰를 파악하여 아키텍처 스팟을 식별하고 설계에서 발생할 수 있는 문제점을 파악하여 해결책을 제시하며 특정 품질 속성이 설계 결정에 미치는 효과에 대해 기술하는 단계이다.

나. 절차

① 작업 1 : 아키텍처 스팟 식별

표현 단계에서 얻어진 다양한 아키텍처 뷰로부터 아키텍처 스팟을 식별한다. 아키텍처 스팟이란 설계에 있어 특정 품질 요소에 영향을 줄 수 있는 아키텍처 뷰나 아키텍처 스타일을 의미한다. 이러한 스팟을 식별하기 위해서는 먼저 시스템의 주요 요구 사항을 기술한 요구 사항 템플릿 분석하고, 템플릿의 기능적 요구에 관련되어 있는 품질 속성을 식별한 후, 이들 품질 속성에 영향을 줄 수 있는 아키텍처 스팟을 파악한다.

② 작업 2 : 아키텍처 설계 결정 식별

아키텍처 설계 결정은 품질 속성 특성표의 원인에 의해 제시될 수 있는 하나 또는 그 이상의 설계 이슈들을 의미하며 품질 속성 획득 여부에 많은 영향을 미치므로 개발 기간 동안 아키텍처 설계 결정의 식별과 분석 작업은 매우 중요하다. 아키텍처 설계 결정을 찾기 위해서는 아키텍처 스팟으로부터 설계상에서 야기될 수 있는 문제점을 식별하여 <표 3>과 같이 결정 변수로 정의한다. 설계상의 지식이나 아키텍처들 간의 비교를 통해 제시된 문제점에 대한 해결 방법 중 한 가지를 결정 값으로 지정하며 다양한 해결 방법 중 결정 값으로 제시되지 않은 나머지를 대안으로 제시할 수 있다.

③ 작업 3 : 아키텍처 설계 결정 분석

식별된 아키텍처 설계 결정들은 하나하나 독립적으로 분석되어야 하며 이들 설계 결정이 특정 품질 속성에 미치는 효과를 원리 항목에 기술하고 관련된 아키텍처 스팟도 함께 표현한다. 이 단계에서의 분석은 상세한 시뮬레이션이나 정확한 수학적 모델링을 필요로 하지는 않는다. 다만 설계에서의 위험 영역을 노출시키기 위한 품질 분석 정도를 의미한다.

다. 산출물

분석 단계의 결과로 나타나는 산출물은 다음과 같다.

- 아키텍처 설계 결정표 : <표 3>

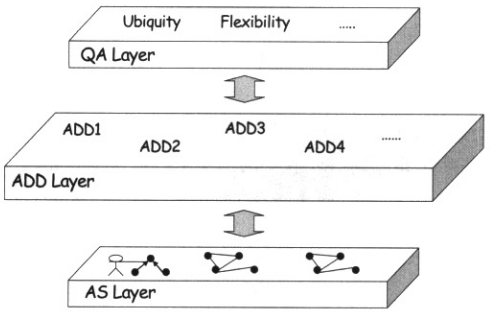
〈표 3〉 아키텍처 설계 결정표

관련 품질 속성	QA1, QA2, . . .		
ADD #	결정 변수	결정 값	대 안
개략적 명세	설계 문제점	설계 해결책	대안 해결책
원리	품질 속성에 미치는 영향		
아키텍처 스팟	해당되는 아키텍처 스팟		

(4) 정제 단계

가. 정의

아키텍처와 품질 속성들 사이의 관계를 이해하여 체계적으로 분류하고, 설계상에서의 위험 요소에 대한 원인과 결과를 파악하며 지금까지의 결과를 기반으로 3-tier 구조를 가지는 종합적인 도메인 아키텍처를 작성하는 단계이다.



(그림 5) 체계도

나. 절차

① 작업 1 : 체계도 작성

이제까지 획득한 산출물 중 품질 속성과 아키텍처 설계 결정, 아키텍처 스팟간의 관계를 도식화하는 단계로서 (그림 5)와 같이 품질 속성 계층, 아키텍처 설계 결정 계층, 아키텍처 스팟 계층으로 구성된다. 각 계층 내에는 기존에 산출된 요소들을 포함하고 있으며, 이들 중 가운데 계층에 있는 아키텍처 설계 결정을 중심으로 품질 속성 계층의 요소들과 설계 결정 요소들 간의 연관관계를 표현하고, 다시 아키텍처 설계 결정과 아키텍처 스팟사이의 연관관계를 표현함으로써 각 계층에 포함되어 있는 요소들 간의 연관 관계를 종합적으로 조직화하게 된다.

아키텍처와 품질 속성들 간의 관계를 체계적으로 보여주는 것은 아키텍처에서 품질 속성을 이해하고 제어하는 능력을 향상시키고 설계에서 위험 영역을 식별하는데 도움을 줄 뿐만 아니라, 자신의 품질 속성에 대한 아키텍처의 타당성을 이해하는데도 유용하다.

② 작업 2 : 아키텍처 위험 명세 기술

잠재적 위험 요소를 가진 설계 결정에 대한 명세를 기술하는 단계로서 <표 4>와 같은 위험 명세표가 산출물로 나타난다. 먼저, 아키텍처 설계 결정으로부터 위험 요인을 식별하여 위험의 종류로 표현하고 관련된 아키텍처 설계 결정을 기술하며 위험에 관련되어 있는 속성을 기술한다. 또한 위험을 야기시키는 구체적인 원인을 기술하고 결과로써 이러한 위험 요소가 품질 속성에 미치는 영향을 파악한다. 마지막으로 이러한 위험에 관련되어 있는 아키텍처 스팟의 식별자를 표기한다.

아키텍처에 관한 위험을 문서로 표현하는 이유는 잠재적인 위험을 노출시킴으로써 아키텍처 설계자가 사전에 위험 요소를 인식할 수 있게 되고 궁극적으로는 위험을 완화시킬 수 있는 계획을 세울 수 있게 하는데 의의가 있다.

〈표 4〉 위험 명세표

RISK #	위험의 종류
품질 속성	관련 있는 품질 속성
아키텍처 설계 결정	관련 있는 아키텍처 설계 결정
원 인	위험의 원인이 되는 상황
결 과	관련된 품질 속성에 미치는 영향

③ 작업 3 : 도메인 아키텍처 작성

지금까지의 결과를 토대로 모바일 시스템의 도메인에 적용될 수 있는 도메인 아키텍처를 작성하는 단계로서 공통의 기능과 요구 기능을 중심으로 개발 환경을 고려하여 3-tier 구조로 도메인 아키텍처를 작성한다.

(그림 6)과 같이 모바일 시스템의 도메인에 적용될 수 있는 도메인 아키텍처는 모바일 시스템에 대한 사용자, 설계자, 개발자의 요구 사항과 기능적 독립성을 고려하여 사용자 계층, 비즈니스 로직 계층, 데이터 계층으로 구성된다. 각 계층은 상호 독립적으로 운용되며 사용자 계층과 비즈니스 계층 간에는 모바일 체제의 네트워크가 존재한다.

- 사용자 계층 : 사용자 측면을 고려하여 화면상에 존재하는 UI를 담당하는 부분으로 다양한 사용자 요구사항을 서비스 형태로 외부에 표현하기 위한 부분이다.
- 비즈니스 로직 계층 : 사용자에게 지원되는 로컬 비즈니스 행위를 구현하는 부분으로 외부적인 서비스를 지원하기 위한 내부 구현과 밀접한 관련이 있는 부분이다. 서비스로는 음성통화, 데이터 전송, E-메일 서비스, 모바일 커머스, 교육이나 게임, 뉴스, 아바타 등과 같은 모바일 컨텐츠, 모바일 뱅킹, 텔레매틱스 등과 같은 모바일 비즈니스가 포함된다.
- 데이터 계층 : 내·외부적인 서비스에서 수반될 수 있는 자료를 관리하기 위한 부분으로 공유되는 자원들에 대한 물리적인 처리를 담당하는 부분이다.

(그림 6) 모바일 시스템 도메인 아키텍처

다. 산출물

정제 단계의 결과로 나타나는 산출물은 다음과 같다.

- 체계도 : (그림 5)
- 위험 명세표 : <표 4>
- 도메인 아키텍처 : (그림 6)

3.2 모바일 학사 시스템 개발

3.2.1 준비 단계

모바일 학사 시스템의 개발 목표, 필요성 및 개발 범위,

스테이크홀더 등을 파악하여 도메인 분석서를 만들고, 요구사항을 수집하여 기능적 요구와 품질 요구를 구분하여 기술할 수 있는 요구 명세 템플릿을 작성한 후 각각의 품질 요구를 상세화하여 품질 속성 명세표를 작성한다.

(1) 도메인 분석

① 시스템 개발 목표

대학교 내 기존 온라인 학사 업무를 모바일로 확장하여 제공함으로써 학사 업무의 효율성을 증가시킬 수 있다. 재사용성 및 유지보수성의 향상을 가져올 수 있는 아키텍처를 구축함으로써 대학 내 학사 업무 비용의 절감 효과를 기대할 수 있다.

② 시스템 개발의 필요성

무선 인터넷 환경의 보급에 따라 학생층을 중심으로 휴대폰 사용자가 급격히 증가하는 추세에 있다. 대학 내 학생들의 정보 활용을 촉진하고 학생과 학교, 학생과 학생간의 학사 업무에 대한 다양한 형태의 커뮤니케이션의 필요성이 대두된다.

③ 개발 대상 및 범위

개발 대상은 대학 내 학사 업무 영역이며 개발 범위는 사용자 인증, 공지 사항 및 학사 일정 관리, 개인 정보 관리, 수강 정보 관리, 성적 정보 관리, 웹 메일 관리이다.

④ 스테이크홀더

- 교수 : 모바일 기기를 사용하여 학사 시스템에 로그온하여 공지 사항과 학사 일정을 조회하고, 개인 정보를 입력하고 수정하고 조회하며, 개설 강좌와 개인 시간표를 조회할 수 있다. 또한 성적을 입력하고 수정하고 조회할 수 있으며 장학정보를 조회한다. E-메일을 송·수신한다.
- 학생 : 모바일 기기를 사용하여 학사 시스템에 로그온하여 공지 사항과 학사 일정을 조회하고, 개인 정보를 입력하고 수정하고 조회할 수 있으며, 개설 강좌를 조회하여 수강 신청을 하고 개인 시간표를 조회한다. 입력된 성적과 장학 정보를 조회할 수 있으며, E-메일을 송·수신한다.
- 직원 : 모바일 기기를 사용하여 학사 시스템에 로그온하여 공지사항과 학사일정을 등록하고 조회할 수 있으며, 개인정보를 입력하고 수정하고 조회한다. 개설 강좌를 조회할 수 있으며, E-메일을 송·수신한다.

(2) 요구사항 수집 및 분석

모바일 학사 시스템은 <표 5>와 같이 6가지 기능적 요구와 11가지 품질 요구를 가진다. 기능적 요구에는 사용자 인증, 공지 사항 및 학사 일정 관리, 개인 정보 관리, 수강 정보 관리, 성적 정보 관리, 웹 메일 관리가 포함되어 있으며 주요 기능에 대한 라벨은 P#으로 부여하였고, 각 기능의 우선순위를 주요 기능의 우측 열에 표현하였다. 즉, 기능적 요구 중 사용자 인증의 우선순위가 가장 높으며 나머지 기능들은 모두 사용자 인증을 성공한 후에 사용 가능한 기능들이다. 이러한 주요 기능들은 필수 기능과 선택 기능으로 나누어져 기술되어 있으며 라벨링 역시 주요 기능에 따른 필수 기능(P1_M1)과 선택 기능(P1_O1)으로 구분하였다. 또한 11가지 품질 요구를 고려할 수 있었으며 QR#로 라벨링 하였다. 각 품질 요구는 하나이상의 기능과 관련될 수 있으며 이들 품질 요구와 주요 기능과의 관련성은 품질 요구 명세

〈표 5〉 모바일 학사 시스템의 요구 명세 템플릿

품질속성	신뢰성(QA1), 즉시성(QA2), 확장성(QA3), 편재성(QA4)	
컨텍스트	신뢰성(QA1)	QR3, QR6, QR7, QR8, QR10, QR11
	즉시성(QA2)	QR3, QR5, QR9
	확장성(QA3)	QR1, QR2
	편재성(QA4)	QR4
기능적 요구 (FRs)	P1 : 사용자 인증 P2 : 공지사항 및 학사 일정 관리 P3 : 개인 정보 관리 P4 : 수강 정보 관리 P5 : 성적 정보 관리 P6 : 웹 메일 관리	1 2 2 2 2 2
	P1_M1 : ID를 입력한다. P1_M2 : 패스워드를 입력한다. P1_M3 : 인증 실패 후 ID와 패스워드 찾기를 한다. P1_M4 : 로그온 정보를 기록한다. P2_M1 : 공지사항을 등록한다. P2_M1 : 공지사항을 등록한다. P2_M2 : 공지사항을 수정한다. P2_M3 : 공지사항을 삭제한다. P2_M4 : 공지사항을 조회한다. P2_M5 : 학사일정을 등록한다. P2_M6 : 학사일정을 수정한다. P2_M7 : 학사일정을 삭제한다. P2_M8 : 학사일정을 조회한다. P3_M1 : 개인 정보를 입력한다. P3_M2 : 개인 정보를 수정한다. P3_M3 : 개인 정보를 삭제한다. P3_M4 : 개인 정보를 조회한다. P4_M1 : 개설 강좌를 조회한다. P4_M2 : 수강 신청을 한다. P4_M3 : 수강 신청을 조회한다. P4_M4 : 개인 시간표를 조회한다. P5_M1 : 성적을 입력한다. P5_M2 : 성적을 수정한다. P5_M3 : 성적을 조회한다. P5_M4 : 장학 정보를 조회한다. P6_M1 : 메일을 작성한다. P6_m2 : 메일을 송신한다. P6_M3 : 메일을 수신한다. P6_M4 : 메일을 임시 보관함에 저장한다.	
	P1_O1 : 로그온 시 ID와 Passwd 저장 여부를 확인한다. P2_O1 : 공지사항을 월별/일별 조회로 선택할 수 있다. P2_O2 : 학사일정을 월별/일별 조회로 선택할 수 있다. P4_O1 : 개설 강좌를 과목 이름 별/교수 이름 별로 조회할 수 있다. P4_O2 : 개설 강좌 조회 후 '신청'키를 사용하여 수강 신청한다.	
품질 요구 (QRs)	QR1 : 다른 모바일 기기에 pluggable 해야 한다.	System
	QR2 : 새로운 기능 추가시 기존 컴포넌트에 대한 재사용 가능성을 파악하여 개발 기간과 비용을 감소하여야 한다.	System
	QR3 : 네트워크 장애로 서비스 장애가 있을 때 백업 장비로 전환하여 계속 수행해야 한다.	System
	QR4 : 해당 지역의 사용자 폭주로 네트워크에 접속되기 어려운 경우 접속 가능 지역을 사용자에게 알려줄 수 있어야 한다.	System
	QR5 : 사용자의 로그온 요청에 대한 시스템의 최대 응답 시간은 3초 이내여야 한다.	P1_M1
	QR6 : 계정에 대한 접근은 인가받은 자만 가능하다.	P1_M2
	QR7 : 사용자 인증이 3번 실패할 때 잠정적으로 회원 자격 정지한다.	P1_M3
	QR8 : 패스워드 변경할 때 기존 패스워드를 한번 더 확인한 후 변경 기능을 제공한다.	P3_M1
	QR9 : 성적을 입력할 때 1분 간격으로 자동으로 저장한다.	P5_M1
	QR10 : 메일을 전송할 때 한번에 실패할 경우 3번까지 재전송을 시도한다.	P6_M1
	QR11 : 최종적으로 전송 실패하였을 경우 임시 보관함에 자동적으로 저장한다.	P6_M4

우측 열에 표현하였다. 모바일 품질 속성의 6가지 범주 내에서 다양한 품질 요구와 품질 속성을 식별할 수 있으나 본 논문에서는 11가지 품질 요구들로부터 신뢰성(QA1), 즉시성(QA2), 확장성(QA3), 편재성(QA4) 4가지 품질 속성만을 고려하였다. 요구 명세 템플릿 작성의 마지막 작업으로 각각의 품질 속성에 연관되어 있는 품질 요구들에 대한 리스트를 컨텍스트 부분에 표현하였다.

(3) 품질 속성 명세화

식별된 품질 속성들을 상세화하는 단계로서 4가지 품질 속성 중 신뢰성에 대한 품질 속성 명세표는 <표 6>과 같다. 구체적 요인으로 보안성과 복구성을 들 수 있고 비인가자의 접근이나 제 3자의 패스워드 변경을 막기 위해서는 지속적인 사용자 접근 감시와 사용자 로그 확보, 패스워드 이중 확인이 이루어져야 함을 알 수 있다.

<표 7>은 즉시성에 대한 품질 속성 명세표로서 응답 시간 적합성과 지속성을 구체적 요인으로 가지며 응답 시간 적합성에 대해 시스템에 대한 신속한 응답요구가 발생할 경우 시스템 알람 호출을 지원함으로써 대응할 수 있다.

<표 8>은 확장성에 대한 품질 속성 명세표로서 구체적 요인에 재사용성과 변경성이 포함되어 있다. 동일 영역의 유사 업무에 대한 개발 요청이 이루어질 경우 공통 컴포넌트 식별과 모듈화를 통해 재사용성을 증가시킬 수 있다.

마지막으로 편재성에 대한 품질 속성 명세표는 <표 9>와 같으며 추적 가능성과 기밀성을 구체적 요인으로 들 수 있고 추적 가능성에 대해 특정 지역에 사용자가 폭주하여 해당지역의 서비스가 이루어지지 않을 경우 AP를 모니터링

〈표 6〉 신뢰성에 대한 품질 속성 명세표

품질속성	요 인	원 인	대 응
QA1 : 신뢰성	보안성	-비인가자 접근 -사용자 인증 실패 -제3자 패스워드변경	-사용자 접근 감시 -사용자 로그 확보 -패스워드 이중 확인
	복구성	-시스템다운 -네트워크다운	-오류 복구, 서버 이중화 -임시데이터 저장

〈표 7〉 즉시성에 대한 품질 속성 명세표

품질속성	요 인	원 인	대 응
QA2 : 즉시성	응답시간 적합성	-신속한 시스템 응답 요구	-시스템 알람 호출
	지속성	-외부 시스템과 연동 불가	-시스템 모니터링

〈표 8〉 확장성에 대한 품질 속성 명세표

품질속성	요 인	원 인	대 응
QA3 : 확장성	재사용성	-동일 영역 유사 업무 개발 요청	-모듈화 -공통 컴포넌트 식별
	변경성	-업무 시스템의 확장 -변경 또는 확장 -요구사항의 변경	-계층화된 아키텍처 -파급 효과 -유연한 시스템 구조

〈표 9〉 편재성에 대한 품질 속성 명세표

품질속성	요 인	원 인	대 응
QA4 : 편재성	추적가능성	-특정 지역 사용자 폭주	-AP 모니터링
	기밀성	-제3자에 의한 데이터 유출	-데이터 암호화

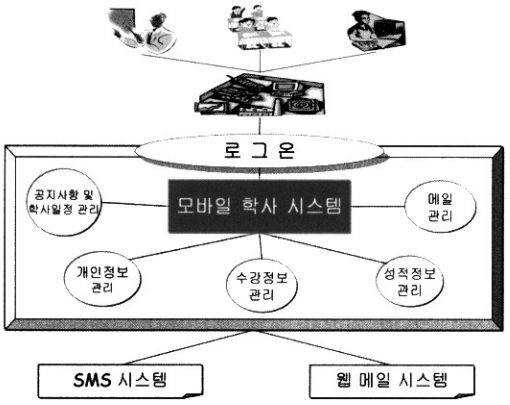
함으로써 적절한 사용자 분배를 도모할 수 있게 된다. 또한 기밀성에 대해 제 3자에 의한 데이터 유출이 발생할 경우 송·수신하는 모든 데이터를 암호화함으로써 기밀성을 유지할 수 있게 된다.

3.2.2 표현 단계

모바일 학사 시스템의 개발 범위와 대상, 스테이크홀더에 대한 내용을 기반으로 시스템의 경계를 나타내는 컨텍스트 뷰를 작성하고 UML을 사용하여 4가지 뷰를 표현한다.

(1) 컨텍스트 뷰

모바일 학사 시스템은 (그림 7)과 같이 교수, 학생, 직원으로 구성된 스테이크홀더들이 모바일 기기를 통하여 가장 우선순위가 높은 기능인 사용자 인증(로그온)을 거친 후에 모바일 학사 시스템의 주요 기능인 공지 사항 및 학사 일정 관리, 개인 정보 관리, 수강 정보 관리, 성적 정보 관리, 웹 메일 관리 등을 이용할 수 있음을 보여주고 있다.



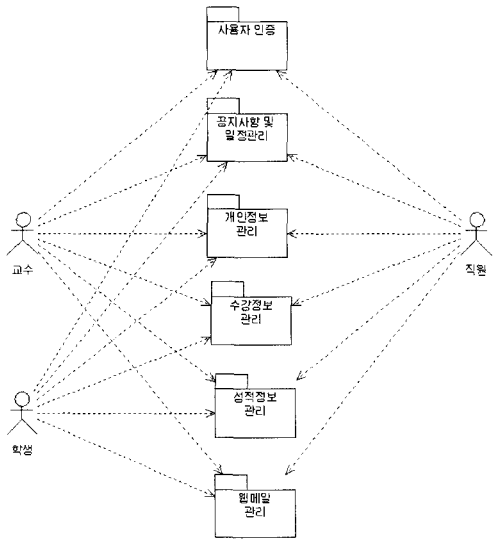
(그림 7) 모바일 학사 시스템의 컨텍스트 뷰

(2) 유스케이스 뷰

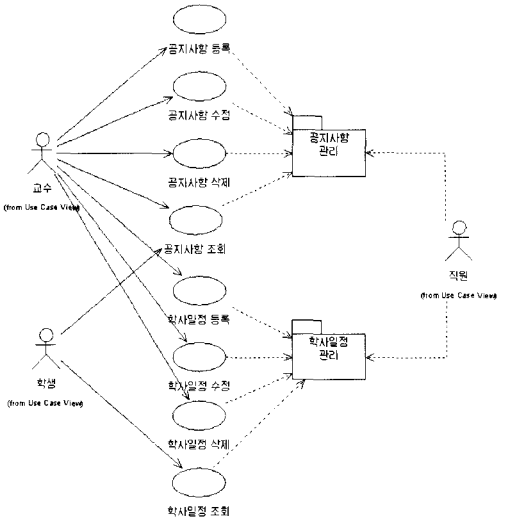
모바일 학사 시스템의 주요 기능들을 패키지 형태로 표현하고 각각의 패키지들에 대한 유스케이스 뷰를 작성하는 단계로서 전체 시스템에 대한 패키지 형태의 유스케이스 뷰는 (그림 8)이며 이 중 공지 사항 및 학사 일정 관리 패키지는 (그림 9)와 같이 나타낼 수 있다.

(3) 논리 뷰

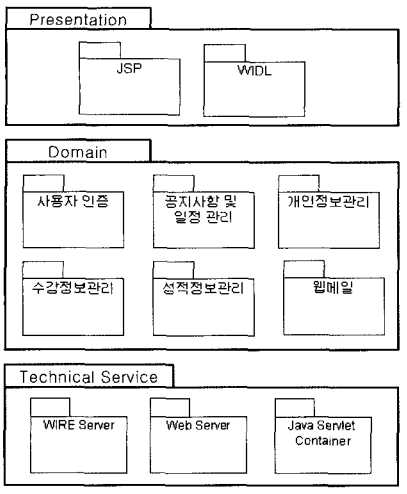
논리 뷰에서는 모바일 학사 시스템의 추상의 집합과 그들의 관계를 보여준다. 시스템의 정적인 측면을 표현한 아키텍처로서 모바일 학사 시스템의 논리 뷰는 (그림 10)과 같이 표현할 수 있다.



(그림 8) 모바일 학사 시스템의 전체 유스케이스 뷰



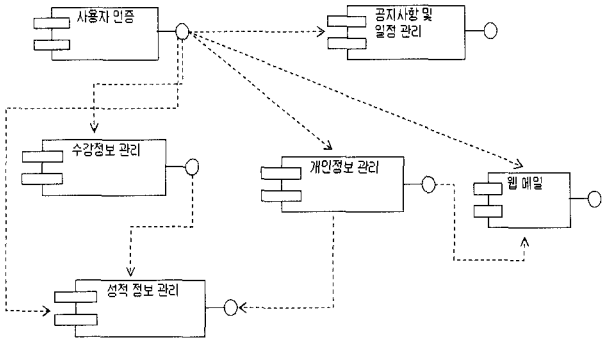
(그림 9) 공지사항 및 학사일정관리 유스케이스뷰



(그림 10) 모바일 학사 시스템의 논리 뷰

(4) 컴포넌트 뷰

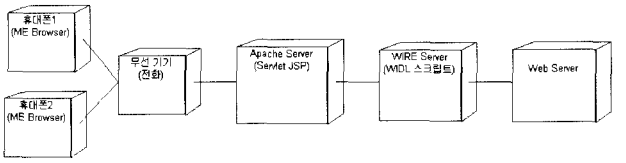
모바일 학사 시스템의 컴포넌트 뷰는 (그림 11)과 같이 시스템에서 식별되는 컴포넌트들 간의 연관관계를 나타내는 아키텍처로서 각각의 컴포넌트들은 사용자 인증 컴포넌트를 통해 다른 컴포넌트들에 연결되어 있음을 알 수 있다.



(그림 11) 모바일 학사 시스템의 컴포넌트 뷰

(5) 배치 뷰

모바일 학사 시스템을 구성하는 물리적인 구성요소들 간의 배치를 표현하는 배치 뷰는 (그림 12)와 같이 나타낼 수 있으며, 웹 서버와 모바일 서버 및 무선기기들로 구성되어 있다.



(그림 12) 모바일 학사 시스템의 배치 뷰

3.2.3 분석 단계

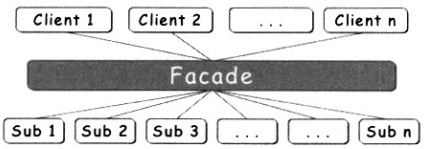
세 번째 단계인 분석 단계는 품질 속성에 연관되어 있는 아키텍처 스팟을 식별하고, 설계 시에 발생할 수 있는 문제점과 해결책 및 대안을 표현하는 아키텍처 설계 결정(ADD : Architecture Design Decision)을 식별하고 분석한다. 본 논문에서는 6가지 아키텍처 스팟을 식별하였고 이들 각각에 관련되어 있는 아키텍처 설계 결정 6가지를 식별하고 분석하였다. 관련된 아키텍처 스팟을 포함하고 있는 6가지 아키텍처 설계 결정표는 다음과 같다.

(1) 계층간 연결 (ADD1)

첫 번째 아키텍처 설계 결정인 계층간 연결은 시스템이 여러 계층으로 나뉘어져 있을 경우 이들 계층들 간의 연결에서 발생할 수 있는 문제를 의미한다. 핵심 문제인 결정 변수 LayersConnection에 대한 해결책인 결정 값으로 Facade의 사용을 제시하였으며 또 다른 해결책 즉, 대안으로는 Controller의 사용과 Observer의 사용을 제시하였다.

계층간 연결의 아키텍처 설계 결정표는 <표 10>과 같다.

〈표 10〉 계층간 연결

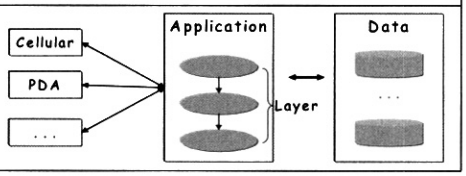
관련 품질 속성	QA1 : 신뢰성, QA3 : 확장성		
Decision : ADD1	결정 변수	결정 값	대 안
계층간 연결	Layers Connection	Using Facade	Using Controller Using Observer
원 리	원격 호출에 의해 야기되는 병목 현상을 방지 서브 시스템들 간의 의존성을 최소화 다른 플랫폼에 시스템을 간단하게 포팅할 수 있는 facade를 가짐		
아키텍처 스팟 : AS1			

(2) 대규모의 논리 구조 (ADD2)

두 번째 아키텍처 설계 결정은 시스템의 논리적 구조가 대규모일 경우에 발생할 수 있는 문제점을 지적하고 있다. 시스템의 논리적 구조가 큰 경우 개발뿐만 아니라 변경에 따른 파급효과가 시스템 전체에 영향을 미칠 수 있다. 그러므로 결정 변수 OverallStructure에 대한 결정 값으로 Layered-Structure를 제시하였으며 대안으로는 Pipeline 사용과 Black-board 사용을 제안하였다.

대규모의 논리 구조에 대한 아키텍처 설계 결정표는 <표 11>과 같다.

〈표 11〉 대규모의 논리 구조

관련 품질 속성	QA3 : 확장성		
Decision : ADD2	결정 변수	결정 값	대 안
대규모의 논리구조	OverlallStructure	Using Layered Structure	Using Pipeline Using Blackboard
원 리	시스템을 계층별로 분리하여 변경에 의한 파급 효과 국지화 향상이나 변경 등 추상화 수준의 증가를 배려하여 설계 호환성 지원		
아키텍처 스팟 : AS2			

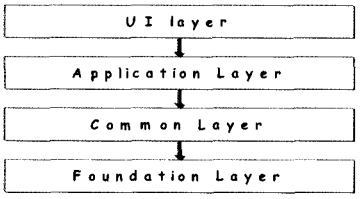
(3) 다양한 규칙 지원 (ADD3)

세 번째 아키텍처 설계 결정인 다양한 규칙 지원은 획일적이지 않은 다양한 규칙들을 지원하기 위해 결정 변수로 RuleSupport를, 결정 값으로 Strategy 사용을 제시하였다.

전략을 통하여 pluggable한 알고리즘이나 규칙의 적용이 가능하며 시스템에서 공동으로 사용하는 모듈을 공통화하여 공통적 요소와 가변적 요소를 분리하여 관리할 수 있으며 변경이나 확장에 용이하게 적용할 수 있게 된다.

다양한 규칙 지원에 대한 아키텍처 설계 결정표는 <표 12>와 같다.

〈표 12〉 다양한 규칙 지원

관련 품질 속성	QA3 : 확장성, QA4 : 편재성		
Decision : ADD3	결정 변수	결정 값	대 안
다양한 규칙지원	RuleSupport	Using Strategy	None
원 리	pluggable한 알고리즘이나 규칙을 적용 시스템에서 공동으로 사용하는 모듈을 공통화하여 공통적 요소와 가변적 요소를 분리하여 관리 변경이나 확장에 용이하게 적용		
아키텍처 스팟 : AS3			

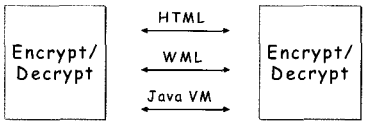
(4) 기밀성 보장 (ADD4)

네 번째 아키텍처 설계 결정인 기밀성 보장은 제 3자에 의한 자료의 유출이나 도청에 관련된 문제를 표현하고 있다. 기존의 웹 애플리케이션에서도 보안에 관한 문제는 중요하게 다루어지고 있으며 모바일 시스템의 특성상 기밀성을 보장하기가 더욱 어려운 것이 사실이다.

이러한 기밀성 보장에 대해 결정 변수 Data Wiretap에 대한 결정 값으로 Encrypt를 제시하였다. 네트워크를 사용하는 모든 데이터에 대해 외부 시스템과의 송·수신 시 데이터를 암호화함으로써 제 3자에 의한 자료의 유출을 방지할 수 있게 된다.

기밀성 보장의 아키텍처 설계 결정표는 <표 13>과 같다.

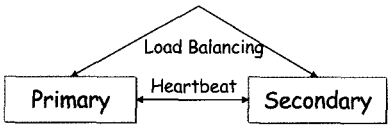
〈표 13〉 기밀성 보장

관련 품질 속성	QA4 : 편재성		
Decision : ADD4	결정 변수	결정 값	대 안
기밀성 보장	DataWiretap	Using Encrypt	None
원 리	네트워크를 사용하는 모든 자료는 외부 시스템과의 송·수신시 암호화 제 3자에 의한 자료 유출 방지		
아키텍처 스팟 : AS4			

(5) 원격 서비스 오류 복구 (ADD5)

다섯 번째 아키텍처 설계 결정인 원격 서비스 오류 복구는 원격 서비스가 실패하였을 경우 복구에 대한 문제를 표현한 것으로 결정 변수 FailureToServices에 대한 결정 값

〈표 14〉 원격 서비스 오류 복구

관련 품질 속성	QA1 : 신뢰성, QA2 : 즉시성		
Decision : ADD5	결정 변수	결정 값	대 안
원격 서비스 오류 복구	Failure ToServices	Using Load Balancing	Using Full Local Replication
원 리	트랜잭션 처리 시에 각 노드의 자원을 공유하여 처리 시스템 환경에 적합한 Load Balancing 알고리즘 적용 원격 서비스가 실패할 때 간단한 서비스는 계속 지원		
아키텍처 스팟 : AS5			

으로 Load Balancing 기법의 사용을 제시하고 대안으로는 Full Local Replication 사용을 제시하였다.

원격 서비스 오류 복구에 대한 아키텍처 설계 결정표는 <표 14>와 같다.

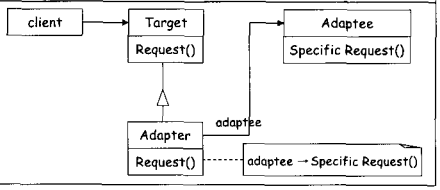
(6) 외부 서비스 지원(ADD6)

마지막 아키텍처 설계 결정인 외부 서비스 지원은 다양한 외부 인터페이스에 대한 서비스 지원에 대한 문제를 나타낸 설계 결정이다. 결정 변수 ProtectVariation에 대한 결정 값으로 어댑터의 사용을 제시하고 대안으로 메시지의 사용을 제안한다.

어댑터는 클래스의 인터페이스를 클라이언트가 기대하는 형태의 인터페이스로 변환해주는 역할을 하는 것으로 어댑터를 통해 외부 서비스의 다양한 인터페이스로부터 변화성을 방지할 수 있으며 메시지 서비스를 사용하는 것보다 안정적인 커뮤니케이션을 지원하지만 메시지 서비스보다는 신뢰성이 떨어진다.

외부 서비스 지원에 대한 아키텍처 설계 결정표는 <표 15>와 같다.

〈표 15〉 외부 서비스 지원

관련 품질 속성	QA3 : 확장성		
Decision : ADD6	결정 변수	결정 값	대 안
외부 서비스 지원	ProtectVariation	Using Adapter	Using Messing
원 리	메시징 서비스를 사용하는 것보다 안정적인 커뮤니케이션 지원 외부 서비스의 다양한 인터페이스로부터 변화성 방지 메시징 서비스보다 낮은 신뢰성을 가짐		
아키텍처 스팟 : AS6			

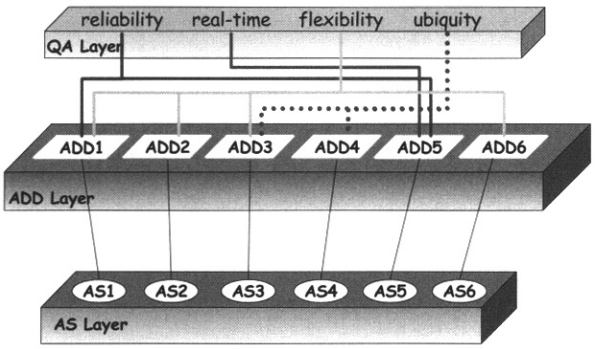
3.2.4 정제 단계

마지막 단계인 정제 단계에서는 모바일 학사 시스템 개발에서 식별된 4가지 품질 속성과 6가지 아키텍처 설계 결정 및 아키텍처 스팟간의 관련성을 나타내는 체계도를 작성하고, 아키텍처 설계 결정으로부터 파악된 위험에 대한 위험 명세표를 작성하며, 전체적인 모바일 학사 시스템의 아키텍처를 작성한다.

(1) 체계도 작성

준비 단계의 요구 사항 분석 및 수집 단계를 통해 신뢰성, 즉시성, 확장성, 편재성 등 4가지 품질 속성을 식별하여 각각에 대한 품질 속성 명세표를 작성하였다. 또한 분석 단계에서는 아키텍처 스팟으로부터 6가지의 아키텍처 설계 결정을 식별하여 분석하였다. 모바일 학사 시스템에서 파악된 품질 속성, 아키텍처 설계 결정, 아키텍처 스팟들 간의 관련성을 나타내기 위해 (그림 13)과 같이 3계층으로 체계도를 작성하였다.

아키텍처와 품질 속성들 사이의 관계를 체계적으로 연결함으로써 특정 품질 속성에 관련되어 있는 아키텍처 설계 결정에 대한 정보를 한 눈에 파악할 수 있으며 품질 속성들 간의 상반 관계를 결정하고 아키텍처 설계를 변경함으로써 야기될 수 있는 효과를 이해하는데 도움이 된다.



(그림 13) 모바일 학사 시스템 체계도

(2) 위험 명세

모바일 학사 시스템의 아키텍처 설계 결정과 체계도로부터 설계 경향이나 결점들을 파악하여 잠재되어 있는 위험을 드러내는 작업이다. 모바일 학사 시스템에서 나타날 수 있는 위험은 응답 시간 효율성과 데이터 전송의 신뢰성 두 가지가 있다. 응답시간 효율성은 대규모 논리 구조와 연관되어 있으며 ADD2에서 제시된 것처럼 계층적 구조를 사용하였을 경우 변경에 따른 파급 효과는 국지화 할 수 있으나, 만일 계층화가 지나치면 계층간의 데이터 전송에 많은 오버헤드가 발생하여 사용자의 요구에 실시간 응답이 어려울 수 있으며 품질 속성 중 즉시성에 영향을 미치게 된다.

응답 시간 효율성에 관한 위험 명세는 <표 16>과 같다.

두 번째 위험 명세는 데이터 전송 중 발생할 수 있는 데이터 신뢰성에 대한 위험으로서 ADD6과 연관되어 있다. 어댑터를 사용함으로써 클래스의 인터페이스를 클라이언트가 기대하는 형태의 인터페이스로 변환하여 데이터를 송·수신

〈표 16〉 응답시간 효율성 위험 명세표

RISK 1	응답시간 효율성
품질 속성	즉 시 성
아키텍처 설계 결정	ADD2
원 인	많은 계층을 통해 사용자의 요구를 전송하는데 오버헤드 발생
결 과	사용자의 요구에 실시간 응답이 어려움

할 수 있게 되지만 어댑터를 거치지 않고 직접 데이터를 송·수신하는 메시징 방법에 비해서는 신뢰성이 떨어진다. 특히, 어댑터를 사용해 RPC(Remote Procedure Call) 스타일의 데이터를 전송할 때 악의적인 공격자에 의한 자료 전송이나 삭제의 위험이 있다.

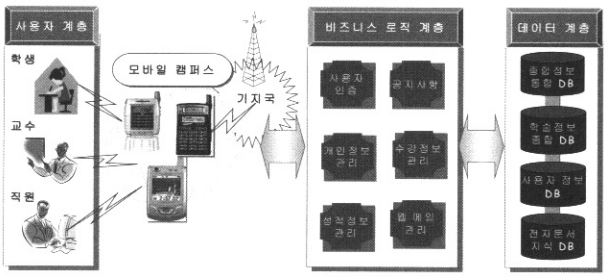
데이터 전송 신뢰성에 관한 위험명세는 <표 17>과 같다.

〈표 17〉 데이터 전송의 신뢰성 위험 명세표

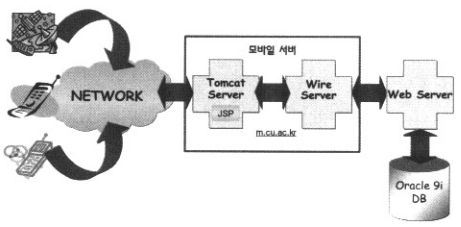
RISK 2	데이터 전송의 신뢰성
품질 속성	신 뢰 성
아키텍처 설계 결정	ADD6
원 인	어댑터를 사용한 RPC 스타일의 데이터 전송
결 과	데이터 전송에 대한 신뢰성 부족

(3) 도메인 아키텍처

모바일 프로덕트 라인 개발 프로세스의 도메인 아키텍처를 기반으로 공통의 기능과 요구 기능을 중심으로 개발 환경을 고려하여 모바일 학사 시스템 도메인 아키텍처를 작성하는 단계이다. (그림 14)와 같이 교육과 업무에 대한 사용자, 개발자, 운영자의 요구사항과 기능적 독립성을 고려하여 3계층으로 구성하고 각 계층은 상호 독립적으로 운용된다.



(그림 14) 모바일 학사 시스템 도메인 아키텍처



(그림 15) 모바일 학사 시스템 구현 환경

3.4 시스템 구현 환경

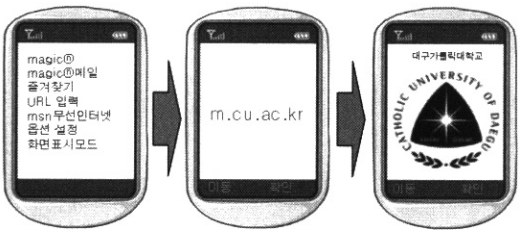
본 논문의 모바일 학사 시스템의 구현 환경은 (그림 15)과 같이 WIRE 서버와 Apache Tomcat 서버가 포함된 모바일 서버, 데이터베이스와 연동되어 있는 웹 서버로 구성되어있다.

모바일 서버에 포함되는 WIRE 서버는 모바일 서버 애플리케이션으로 클라이언트의 요청에 의해 웹 서버상의 애플리케이션을 호출하여 결과를 클라이언트에게 전달하는 역할을 한다. Apache Tomcat 서버는 Java Servlet Container로서 Java Servlet을 구동할 수 있으며, 실제로 휴대폰의 ME Browser로 페이지를 전송한다. Java Servlet은 WIRE 서버와 통신하여 휴대폰의 페이지를 생성하는 코드로 WIRE 서버의 직접적인 클라이언트가 된다.

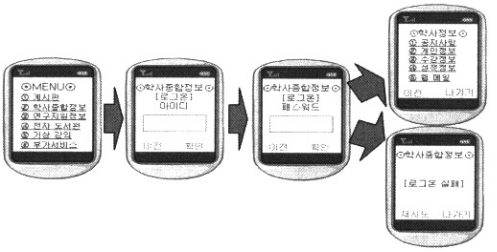
3.5 실행 예

(그림 16)은 모바일 홈페이지의 접속 화면에 대한 실행 예이다. 먼저 사용자가 휴대폰의 인터넷 접속 버튼을 누르고 'URL 입력'을 선택하여 원하는 URL을 입력하고 확인 버튼을 누른 후 모바일 홈페이지에 접속하는 화면이다.

(그림 17)은 모바일 학사 시스템에 접속하기 위하여 사용자 인증을 진행하는 화면이다. 먼저 대학 내 모바일 홈페이지에 접속하여 학사 종합 정보를 선택하여 사용자 인증을 위해 아이디를 입력하고 확인한 후 패스워드를 입력한다. 시스템이 보유하고 있는 사용자 정보와 동일할 경우에는 학사정보 초기화면을 제공해주고 동일하지 않을 경우에는 로그 온 실패를 알리는 화면을 제공해준다.



(그림 16) 모바일 홈페이지 접속 화면



(그림 17) 모바일 학사 시스템의 로그온 화면

4. 결 론

본 논문에서는 CBD 기반의 소프트웨어 프로덕트 라인을

도입하여 모바일 응용 시스템에 적용한 것으로 영역 공학에서 식별된 모바일 프로덕트 라인 아키텍처를 기반으로 핵심자산들을 핵심자산 저장소에서 관리하고 응용 공학에서 재사용하여 프로덕트인 모바일 학사 시스템을 개발하였다. 이를 위해 먼저 영역 공학과 관리 및 응용 공학을 모두 포함하는 모바일 프로덕트 라인 개발 프로세스를 정의하고 영역 공학 단계로 모바일 응용 시스템의 요구사항을 분석하여 모바일 응용 시스템 아키텍처 개발 프로세스를 제안하였다.

제안된 프로세스는 아키텍처 개발을 위해 준비, 표현, 분석, 정제 단계로 구성된다. 준비 단계에는 목표 시스템의 개발 범위와 목표 등을 파악하는 도메인 분석과 요구사항 수집 및 분석, 품질 속성 명세화 작업을 제시하였다. 표현단계는 시스템의 경계를 표현하는 컨텍스트 뷰와 요구사항으로부터 유스케이스 뷰, 논리 뷰, 컴포넌트 뷰, 배치 뷰를 작성한다. 분석 단계는 아키텍처 뷰로부터 설계에 영향을 줄 수 있는 뷰들을 파악하여 아키텍처 스팟으로 식별하고 이들 아키텍처 스팟으로부터 아키텍처 설계 결정을 식별하고 분석하여 아키텍처 설계 결정표를 제시하였다. 마지막 정제 단계에서는 품질 속성과 아키텍처 설계 결정, 아키텍처 스팟간의 관계를 파악할 수 있는 체계도를 작성하고 잠재적인 위험을 명세하며 지금까지의 결과를 바탕으로 시스템의 공통 기능과 요구 기능을 중심으로 개발 환경을 고려한 도메인 아키텍처를 작성하였다. 이와 같이 각 단계마다 세부적인 작업 절차와 산출물을 제시하였으며 이들 세부 작업을 수행함에 따라 품질 속성 중심의 체계적인 아키텍처 개발이 가능하였다.

응용 공학 단계에서는 개발된 모바일 응용 시스템 아키텍처를 대학 내 학사 시스템에 적용함으로써 모바일 학사 시스템을 개발하였다. 모바일 학사 시스템 컴포넌트와 일반 모바일 응용 시스템 구축에서 요구되는 서비스 모듈들을 ABCD 참조 아키텍처에 기반하여 모바일 시스템 영역에서 요구되는 컴포넌트 계층을 정의하였다.

기대 효과로는 소프트웨어 핵심자산의 체계적인 개발과 관리 및 조직적인 재사용을 통하여 모바일 응용 시스템 개발 기간과 비용을 줄일 수 있고, 검증된 핵심자산을 사용함으로써 소프트웨어 품질 보장과 다양한 요구사항을 쉽고 빠르게 반영할 수 있다. 또한 품질 속성 중심의 아키텍처 설계 결정을 제시하고 잠재적인 위험을 명세함으로써 아키텍처의 풍부한 재사용 가능성을 제공할 수 있게 된다. 특히 모바일 시스템과 같은 실시간 응답이 요구되는 응용 시스템에서 소비자 요구를 충족시킬 수 있으므로 더욱 효율적이다.

향후 연구로는 동일 영역 프로덕트 생산 시 핵심자산들을 재사용하여 소프트웨어 프로덕트를 조립 및 생산하는 기술에 관한 연구와 다양한 영역별 프로덕트 라인 아키텍처 개발을 통한 특정 프로덕트로의 적용이 필요하다.

참 고 문 헌

[1] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere,, "The Architecture Tradeoff Analysis Method," The 4th IEEE International Conference on Engineering of Complex Computer Systems, pp.68-78, August, 1998.

[2] Dobrica, L. and Niemela, E., "A Survey on Software Architecture Analysis Engineering" IEEE Transactions on Software Engineering, Vol.28, No.7, pp.638-653, July, 2002.

[3] Klaus Schmid, "The Economic Impact of Product Line Adoption and Evolution," IEEE SOFTWARE, Vol.19 No.4, pp.50-57, July/August, 2002.

[4] Eila Niemela, Tuomas Ihme, "Product line software engineering of embedded systems," Symposium on Software Reusability Proceedings of the 2001 symposium on Software reusability, putting software reuse in context, Toronto, Ontario, Canada, pp.118-125, 2001.

[5] Northrop, "A Framework for Software Product Line Practice," http://www.sei.cmu.edu/plp/framework.html, 2001.

[6] Klaus Schmid, "People Issues in developing Software Product Lines," IESE-Report No. 051.01/E, Version 1.0, 2001.

[7] Colin Atkinson, Component-based Product Line Engineering with UML, Addision-Wesley, 2002.

[8] Mari Matinlassi, "Comparison of Software Product Line Architecture Design Methods : COPA, FAST, FORM, KobrA and QADA," International Conference on Software Engineering, Proceedings of the 26th International Conference on Software Engineering, Vol.00, pp.127-136. 2004.

[9] Charles W. Krueger, "Variation Management for Software Product Lines," SPLC 2, San Diego, CA, USA, Vol.2379, pp. 37-48, 2002.

[10] 김기천, "모바일 서비스 기술 동향," 한국정보처리학회지, 제9권, 제2호, pp.17-23, March, 2002.

[11] Jacobson, Booch, Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999.

[12] Christine Hofmeister, Robert Nord and Dilip Soni, Applied Software Architecture, Addsion-Wesley, 2000.

[13] Carnegie Mellon University, "How Do You Define Software Architecture," February, 2003. http://www.sei.cmu.edu/architecture/definitions.html,

[14] http://www.component.or.kr/Architecture/ 2003Architecture_Framework.zip

[15] The Open Group Architecture Framework(TOGAF) Ver 6, The Open Group, 2000.

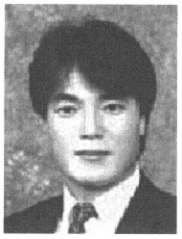
[16] Haeng-Kon Kim, Lee-Kyeong Son "A Study on Software Architecture Evaluation," Journal of Electronics & Computer Science, Vol.5, No.2, pp.37-48, 2003.

[17] Action Semantics Models, Unified Modeling Language Specification, Version 1.5 OMG Document, Formal /03-03-01, 2003.

[18] Andreas Hein, "Systematic Integration of Variability into Product Line Architecture Design," SPLC 2, San Diego, CA, USA, Vol.2379, pp.130-153, 2002.

[19] Steffen Thiel, "Modeling and Using Product Line Variability in Automotive Systems," IEEE SOFTWARE, Vol.19 No.4, pp.66-72, 2002.

[20] Carnegie Mellon University, "Domain Engineering and Domain Analysis," http://www.sei.cmu.edu/str/descriptions2002.



김 행 곤

e-mail : hangkon@cu.ac.kr

1985년 중앙대학교 전자계산학과(공학사)

1987년 중앙대학교 대학원 전자계산학과 (공학석사)

1991년 중앙대학교 대학원 전자계산학과 (공학박사)

1978년~1979년 미 항공우주국 객원 연구원

1987년~1989년 한국전기통신공사 전임연구원

1988년~1989년 AT&T 객원 연구원

2001년~2002년 Central Michigan University 교환교수

1990년~2000년 대구효성가톨릭대학교 컴퓨터공학과 부교수

2000년~현재 대구가톨릭대학교 컴퓨터공학과 교수

관심분야 : 컴포넌트기반 소프트웨어공학, 임베디드 소프트웨어 개발 방법론 및 툴, 프로덕트라인 공학

손 이 경

e-mail : twilly@cu.ac.kr

1991년 효성여자대학교 수학교육학과 (이학사)

1993년 효성여자대학교 전산통계학과 (이학석사)

2005년 대구가톨릭대학교 전산통계학과 (이학박사)

현 재 대구가톨릭대학교 컴퓨터정보통신공학부 누리(NURI)전담 교수

관심분야 : 컴포넌트기반 소프트웨어공학, 임베디드 소프트웨어 개발 방법론 및 툴, 프로덕트라인 공학